Lab 2 (Unit Testing) – Assessed Lab

Nipun Jasti

Njas187

Task 1:

- Downloaded the three files from canvas.
- Created test_add in the file test_maths, where I arrange, assert. It was straight forward and checked if the test case works for add function.
- Created test_Fibonacci in the file test_maths, which checks the Fibonacci function where it arrange, act, assert. It was straight forward and checked if the test case works for Fibonacci function. However I found an error in the Fibonacci function in main file (maths). The return at the end of the function it returns first = 0 and second = 1 into a function 'internal'. In internal if count is not equal to 0 then there is a recursion in internal where it returns second, third and count back into itself until count is equal to 0. However to get Fibonacci in this program you need to return second as 0 so I just changed the initial first = 1 and second = 0.

From this task I learned how to unit test a file.

Task 2:

- Created test_convert_base1 in the file test_maths, which converts a base 10 number to any base number where it arrange, act, assert. It's sends a test case into the function in maths file and checks the output. I found an error in the initial code where current at the end of the code is calculated by dividing the number. This was fine however the symbol used was current = current / n which for example current = 5 and n = 2 then 5 / 2, current will equal 1.5 however its meant to round so I changed it to current = current // n which rounds the answer if its not a whole number.
- I added an optional parameter called base, which is set as a default as 10. If you want to add two number and get the base of the number then you put in then you can use this parameter. I then created test_optional_parameter_add in the file test_maths, where it arrange, act, assert. It input two number into the add function like test_add but it also inputs a base. It was straight forward and checked if the test case works with the new parameter.

Form this task I learned how to fix code errors and add more options.

Task 3:

- The methods that I decided to test are the __init__, info and error.
- Created test_logger file which uses unit testing to test the python file with the dictionary unittest.
- Created test_init in the file test_logger, which tests if the class was initialise where it arrange, act, assert. It was straight forward and checked if the test case works.
- Created test_info in the file test_logger, which tests if the function info in the class Logger. It was straight forward and checked if the test case works.
- Created test_error in the file test_logger, which tests if the function error in the class Logger. It was straight forward and check if the test case works.

This lab has helped me understand software development, by showing how to test any section of code to see if it functions the way I wish it to function in.